Towards Automatic Triggering of Android Malware

Adrien Abraham, Radoniaina Andriatsimandefitra Ratsisahanana, Nicolas Kiss, Jean-François Lalande, Valérie Viet Triem Tong

Towards Automatic Triggering of Android Malware

A. Abraham R. Andriatsimandefitra N. Kiss J.-F. Lalande V. Viet Triem Tong

INRIA/CENTRALESUPELEC research project Cidre – INSA CVL
France

The problem: malware hiding techniques

Malware wait before running to evade dynamic analysis

- a fixed or dynamic period of time
- a user input
- a system event
- an order from a remote server
- a particular state of their hosted application
- something else ?

Existing solutions

Some frameworks proposed to test the infected application

1. using random inputs
2. running a maximal branches of code

BUT

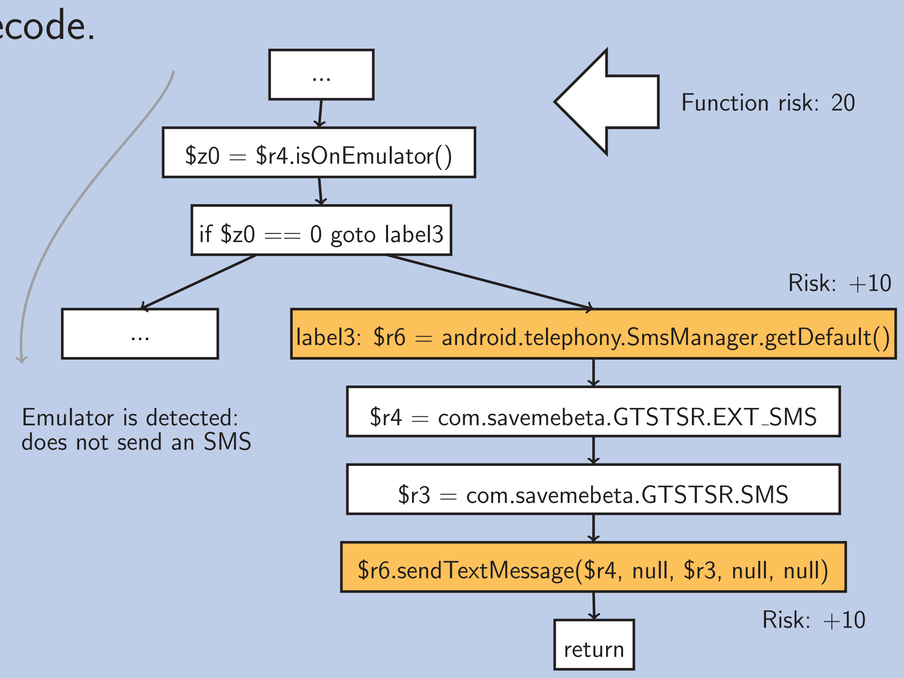
1. it is uncertain
2. it is unnecessarily expensive

First step: static identification of malicious code

A scoring function computes an indicator of risk for each instruction in the bytecode.

The score increases with calls to specific Java methods such as:

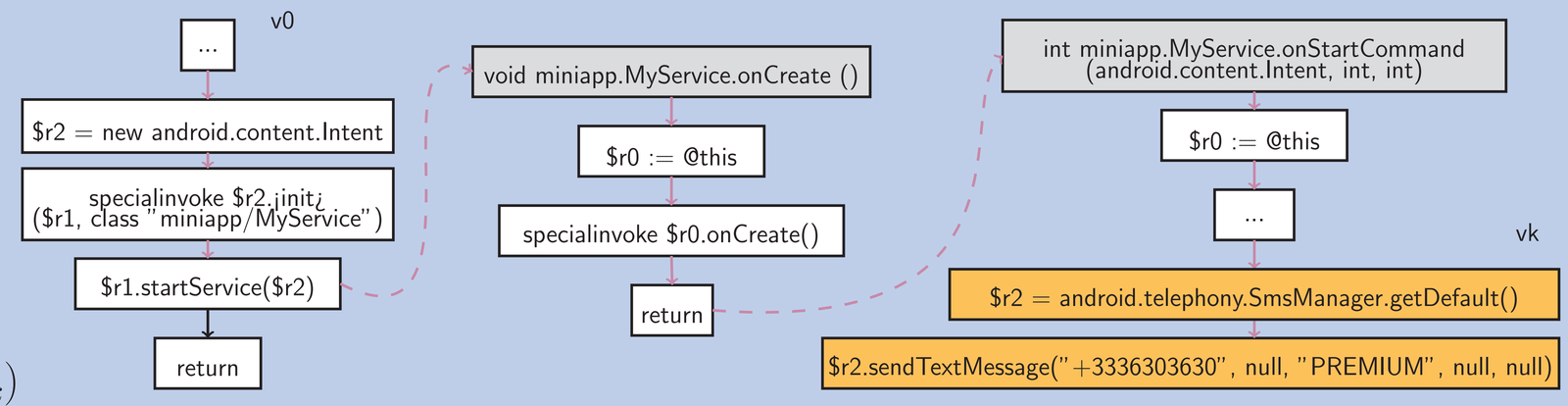
- `android.telephony.SmsManager` for sending SMS
- `android.telephony.TelephonyManager` for getting device infos
- `android.context.pm.PackageManager` for installing/removing apps
- `java.util.Timer,TimerTask` for the implementation of *timebombs*
- `java.lang.Runtime,Process` for executing native binaries
- `dalvik.system.DexClassLoader` for loading code dynamically



Second step: recomputing an execution path to the identified malicious code

To compute an execution path to the most scored unit of code:

- $\forall$ f, functions of the malware, compute: $G_f = (V_f, A_f)$. Let $G = \cup_f G_f = (V, A)$
- $\forall$ intents, events from $v_i \in V_i$ to $v_j \in V_j$: Add (v_i, v_j) to A
- Let v_k the scored unit of code
  Let v_0 the entry point (onCreate())
  Compute $path = shortest\_path(G, v_0, v_k)$



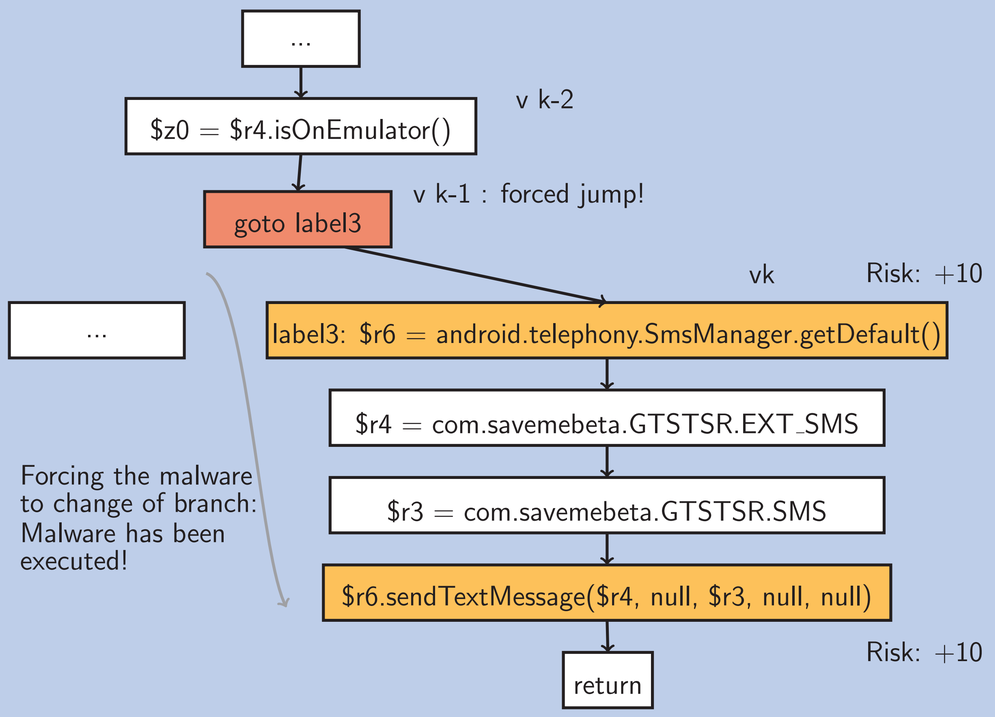
Third step: forcing the execution path

To force the execution of the most scored unit of code $path$:

- Make a *standard* execution
- Let $path = (v_1, ..., v_e, ..., v_k)$:
  v_e is the last unit of code executed.
- $\forall i > e > k$, if v_i is a condition, $Force(v_i)$
- Execute the malware again.

Benefits:

- a malware that is executed shows its effects
- detection tools can be trained or evaluated



Contact: valerie.viettriemtong@centralesupelec.fr